WHAT IS CLAIMED IS:

1. A system for a generic parser, comprising:

a parser communicatively coupled with layered components, the layered components requiring data passed between logical layers of a computing system for communication with platform hardware and platform firmware;

a data buffer having data communicated between the layered components, the data buffer having a data structure comprehensible to the parser; and

layered component drivers comprehending the data buffer after conversion by the parser.

2. The system as recited in claim 1, wherein the communicated data buffer is defined by a buffer map, the buffer map comprising a type field, a length field and at least one field defining data types corresponding to data in the communicated data buffer.

3. The system as recited in claim 2, wherein the buffer map corresponding to data buffer are communicated separately.

4. The system as recited in claim 2, wherein each field defining data types is one of a derived data type and base data type.

5. The system as recited in claim 4, wherein a derived data type comprises at least one of a derived data type and a base data type.

6. The system as recited in claim 4, wherein a common set of base data types are defined in terms of size, format and semantic meaning.

7. The system as recited in claim 2, wherein the parser uses a hash table to search a derived type table and base type table to fully comprehend the communicated data buffer in terms of contained data instances, their format and semantic meaning.

8. The system as recited in claim 1, wherein an in-line data byte order of the data buffer is transformed to and from big endian and little endian byte order.

9. The system as recited in claim 1, wherein pointers are converted in-line based on a new buffer base address.

10. The system as recited in claim 1, wherein pointers are converted in-line to new address space after crossing virtual memory boundaries.

11. The system as recited in claim 1, wherein data type translation is performed to comprehend the communicated data buffer between layered components.

12. The system as recited in claim 1, wherein conversion of data from one format into another is performed to comprehend the communicated data buffer between layered components.

13. The system as recited in claim 1, wherein transformation of data units is performed to comprehend the communicated data buffer between layered components.

14. The system as recited in claim 1, wherein data is scaled via mathematical transformation to comprehend the communicated data buffer between layered components.

15. A method for using a communicated data buffer among logical layered components in a computing system, comprising:

reading a unit of data from the communicated data buffer, by the parser, according to a buffer map;

determining a type associated with the unit of data;

if the unit of data type corresponds to a derived data type, then identifying data structures comprising the derived data type; and

communicating with at least one layered component, wherein the layered component requires access to a structure in the communicated data buffer.

16. The method as recited in claim 15, wherein the buffer map is communicated separately from the data buffer.

17. The method as recited in claim 15, wherein identifying data structures further comprises:

extracting a data type map identifier from the unit of data;

searching for the data type map identifier in a table of derived types; and

if the data type map identifier is not found in the table of derived types, then searching for the data type map identifier in a table of base types,

wherein extracting and searching continue until all data map identifiers in the data unit have been extracted to their base types.

18. The method as recited in claim 15, wherein if the unit of data type is a data buffer type, then extracting data from the communicated data buffer based on a data type map associated with the communicated data buffer.

19. The method as recited in claim 18, wherein the extracted data corresponds to a data structure required by the layered component.

20. The method as recited by claim 15, wherein the layered component is a virtual interface layer.

21. The method as recited by claim 20, wherein data type maps are defined for a plurality of hardware components, and wherein the parser enables the virtual interface layer to communicate with an existing hardware component.

22. The method as recited by claim 21, wherein the virtual interface layer publishes a hardware interface that does not correspond to a defined data type map, wherein a map is defined for translating the published hardware interface to data type comprehended by the parser.

23. The method as recited in claim 15, wherein an in-line data byte order of the data buffer is transformed to and from big endian and little endian byte order.

24. The method as recited in claim 15, wherein pointers are converted in-line based on a new buffer base address.

25. The method as recited in claim 15, wherein pointers are converted in-line to new address space after crossing virtual memory boundaries.

26. The method as recited in claim 15, wherein data type translation is performed to comprehend the communicated data buffer between layered components.

27. The method as recited in claim 15, wherein conversion of data from one format into another is performed to comprehend the communicated data buffer between layered components.

28. The method as recited in claim 15, wherein transformation of units of data is performed to comprehend the communicated data buffer between layered components.

29. The method as recited in claim 15, wherein data is scaled via mathematical transformation to comprehend the communicated data buffer between layered components.

30. An article of manufacture comprising a machine accessible medium containing code having instructions that, when executed during pre-boot, cause the machine to:

read a unit of data from the communicated data buffer, by a parser, according to a buffer map;

use a buffer map to determine data types contained within the communicated data buffer;

use a data type map to identify data structures contained within the communicated data buffer; and

communicate with at least one layered component, wherein the at least one layered component requires access to a structure in the communicated data buffer.

31. The article as recited in claim 30, wherein , the buffer map is communicated separately from the data buffer.

32. The article as recited in claim 30, wherein identifying data structures further comprises code causing the machine to:

extract a data type map identifier from the unit of data;

search for the data type map identifier in a table of derived types; and

if the data type map identifier is not found in the table of derived types, then search for the data type map identifier in a table of base types,

wherein extracting and searching continue until all data map identifiers in the data packet have been extracted to their base types.

33. The article as recited in claim 30, wherein if the data unit type is a data buffer type, then further comprising code with causes the machine to extract data from the communicated data buffer based on a data type map associated with the communicated data buffer.

34. The article as recited in claim 33, wherein the extracted data corresponds to a data structure required by the layered component.

35. The article as recited by claim 30, wherein the layered component is a virtual interface layer.

36. The article as recited by claim 35, wherein data type maps are defined for a plurality of hardware components, and wherein the parser enables the virtual interface layer to communicate with an existing hardware component.

37. The article as recited by claim 36, wherein the virtual interface layer publishes a hardware interface that does not correspond to a defined data type map.